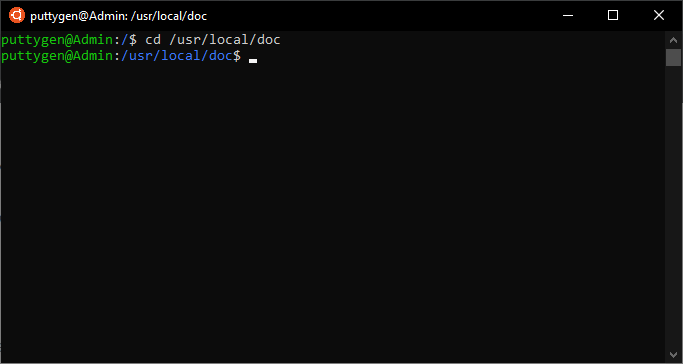
**1. cd Command**

The ‘cd’ is one of the most common and [basic commands used on Linux](https://www.puttygen.com/putty-commands). The primary purpose of the command is to alter the current directory. In simple words, its primary goal is to shift users to a new location in the file system. So, programmers can use ‘cd’ to change to a directory that is within the current directory. Users are required to type cd and the name of the other directory. For example, to go to the directory named ‘office,’ users can type - cd office.



In case, a user wants to move to a directory in the filesystem; then they need to enter the path of the directory. Users must remember to add a leading slash (/) when entering the path. Below is an example:

cd /usr/local/doc

Now to go back to the home directory, a coder has to input the ~ (tilde) character. So the command will look like cd ~.

Let’s view all commands in progressive order:

smith@puttygen: ~$ cd officesmith@puttygen: ~/office$ cd /usr/local/docsmith@puttygen: /usr/local/doc$ cd ~smith@puttygen: ~$

Users can also choose to use . . (double dots) to state the parent of the directory in use. Many use the double dot symbol to shorten the command. Below is an example of the same:

cd ../office

The cd command is very popular among Linux users. For example, users can type**‘cd Downloads’** to move from the home folder to the Downloads folder.

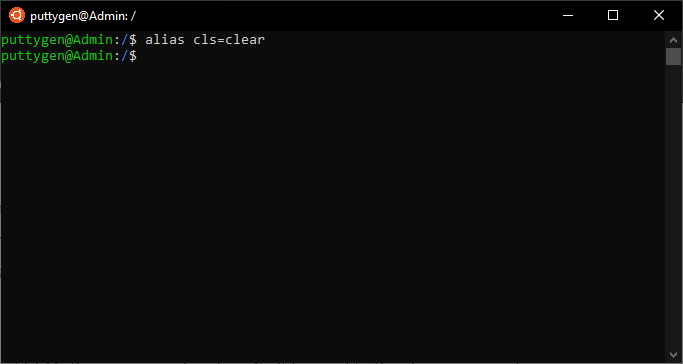
Note: Programmers be cautious as the command is case sensitive and the user has to type exactly the same name as of the folder.

The cd command has a downside. Let’s say the computer has a folder named ‘Raspberry Pi.’ In this case, the command cd considers hat second argument independently, hence resulting in an error. For such instance, coders can use the backward slash as a separator. See the example for the same:

cd Raspberry\ Pi

**2. Alias Command**

The command Alias is an amazing way to personalize and organize all your commands. It allows users to designate a name to a single command or even a string of commands. So programmers can give a short name before executing it. Here is an example that uses an alias:

alias cls=clear

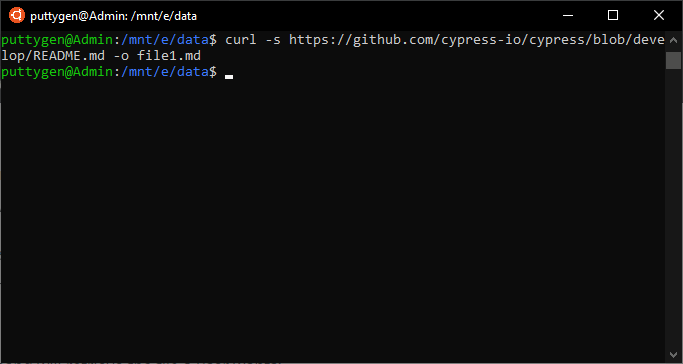
The above command indicates that ‘cls’ is the new name for clear. So it is equal to the clear command, so whenever a user types cls, it will clear the screen. One of the main purposes of an alias is to save time, especially when some default command names are long. More so, personalizing commands helps when switching between operating systems such as from Windows to Linux.

Also, programmers can use an alias if they frequently misspell. For instance, if you type ‘acb’ while the actual command is ‘abc.’ Therefore programmers can execute the following command, which will make sure it takes both spellings as true:

alias acb=“abc”

**3. curl Command**

The curl is a highly functional tool to recover data from URLs (Uniform Resource Locators) or internet repositories. The tool is not available by default on many [Linux distributions](https://www.puttygen.com/linux-distros). For Ubuntu or Debian-based Linux variations, use apt-get to install the package. Other distributions have to use the package managers. Package managers terminate the need to install software and instead makes it automated manually. Below is the expression to install on Debian-based Linux variations:

sudo apt-get install curl

Now, let’s look at the use of curl command. For instance, if a user wants to retrieve a particular file from the GitHub repository. While there is no official way that GitHub provides but instead users are forced to clone the entire database. However, using **curl command**, you can quickly recover a single file.

To retrieve a file, users have to state the exact file name to save it in. Programmers can leverage the **-o (output) option**. Note: If a user does not use the -o option in the command, then the files will not be saved on the computer.

The following single line of command will retrieve the file a user wants:

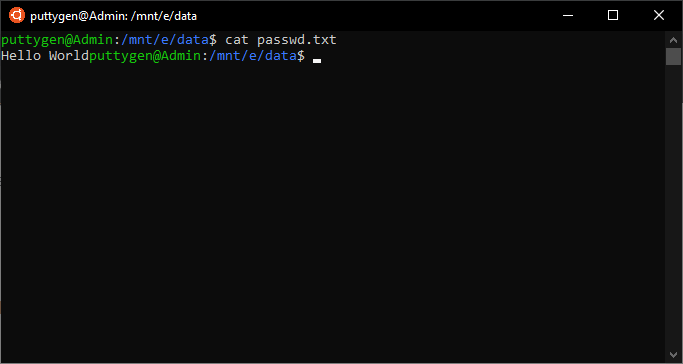
curl https://raw.githubusercontent.com/smiths/linux/master/kernel/events/core.c -o core.c

To avoid viewing the downloading progress or in other words hide it, users can use the -s (silent) option. So the new expression will look as below:

curl -s https://raw.githubusercontent.com/smiths/linux/master/kernel/events/core.c -o core.c

**4. cat Command**

The cat command is simple, and one of the most frequently used commands that are used in a variety of ways. It is short for ‘concatenate’ and permits users to create files, redirect output, list the contents of a file, and even concatenate multiple files.



The command cat is convenient and allows the user to accomplish many tasks with a single expression. For example, to view the content of the /abc/passwd file from the home directory, type the below command:

cat /etc/passwd

Next, type the following to view the contents of multiple files, in this case, test2 and test3:

cat test2 test 3

Now in case the content of a file is long and does not fit the terminal, then the user can use the following parameters:

cat test.txt | more

Now, if a user wants to see less, a user can go back by typing the below parameter:

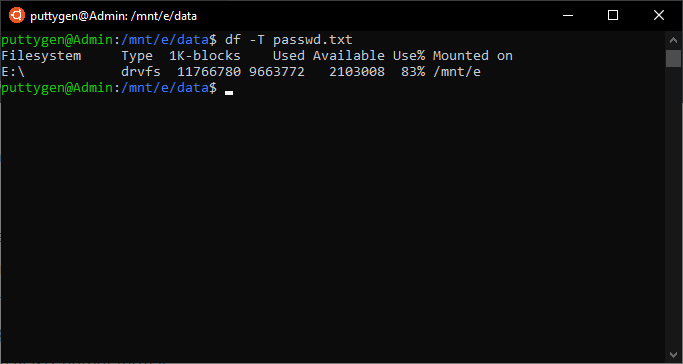
cat tes.txt | less

Coders can also leverage the cat command to view several lines in a file whose content is on the output terminal. To do so, a programmer has to use -n option.

cat -n test.txt

**5. df Command**

The df is a fundamental command on Linux. Using df, coders can display the size, available space, and used space on the filesystems of the device. It comes with multiple options, such as the following:

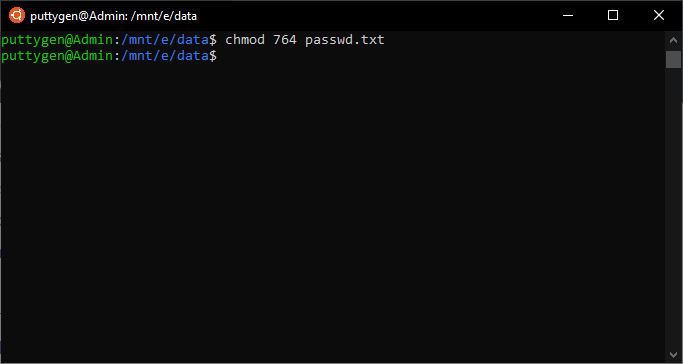


* -a, –all : it includes duplicate and files that are inaccessible
* -x: exclude specific filesystems
* -i, –inodes: it list inode information rather than that of block usage
* -T, –print-type: prints the type of file system
* -P, –portability: it will use the POSIX output format
* -B, –block-size=SIZE: increase the size before printing the result
* -h, –human-readable: print sizes in power of 1024, which is Mb or Gb
* -H, –si: shows result in the power of 1000
* –total: omits all entries not relevant to available space, and presents the total
* -t, –type=TYPE: the result is limited to listing to file systems of a particular type
* –sync: invoke sync before the usage info
* -l, –local: limit listing to local file systems

The -h(human-readable) option displays size in Mb or Gb, rather than bytes. Meanwhile, the -x(exclude) option allows uses to discount specific filesystems.

**6. chmod Command**

The chmod command allows users to alter or assign file permissions flags on a file or folder. The permission defines who can read, write, or run the file. The code comes with references, which indicates to whom the permission will be applicable.



Specific letters represent the references, and programmers can use these letters to assign permission. Below is the list of letters:

* u - the owner of the file
* g - this represents a group. So when assigned, it will apply to all members of the file’s group.
* o - users who are not the owner or a group member
* a - those who are all of the above

Then comes the ‘modes’ that represent each of the permission types. The following are the permissions that a user can assign for a file:

* r - the letter means the user has the permission to the only read
* w - the letter corresponds to write or delete the file
* x - the mode allows users to execute a file

Users can view permissions of all files using the -l command. It will list file in the current directory. It will result in a string of characters as follows:

-rwxrwxrwx

If its a file the output starts with ‘-’ while for a directory it begins with.’

A series of three letters represent each reference. So the first corresponds to the owner, followed by the ‘group’ and lastly for ‘others.’

So, one can use chmod to give change or assign permission to either an owner, group, or others. More so, the string of characters is followed by digits which represent the specific type of file permission.

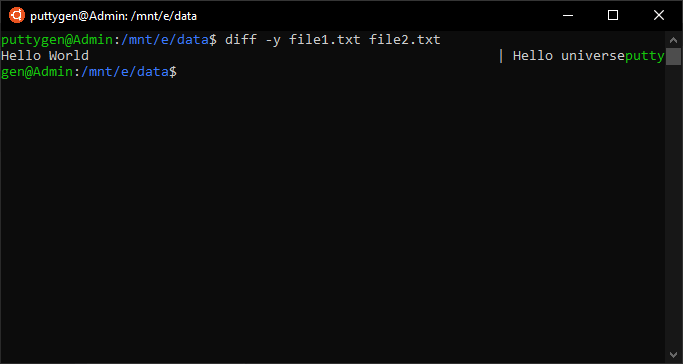
* 0: No permission is assigned
* 1: Execute permission
* 2: it has to write permission
* 3: the user has write and execute permissions
* 4: users can only read with this permission
* 5: users have read and permission to execute
* 6: it indicates that you have both read and write permissions
* 7: it offers you to do anything with the file such as read, write and execute

Looking at our example.txt file, we can see that all three sets of characters are rwx. That means everyone has read, write, and execute rights with the file. Programmers can use the digits with the chmod command to assign permission. For example, to give the owner complete permission - read, write and execute, to provide a group read and write permits and read-only permission to others, enter the following command:

chmod 764 test.txt

**7. diff Command**

The diff command comes in very handy for those that deal with organizing a lot of data. The command will compare data between two text files and display the difference. Coders can work in many ways to customize how the data is displayed.



The command comes with two options, which are -y and -w. The -y is a way to display the difference between two files side by side. Meanwhile, the -w is the option to decide the maximum line width to ensure the lines don’t need wraparound.

Programmers can use --suppress-common-line to prevent the diff from listing the same lines. Therefore, you can put their attention to only view the lines that are different.

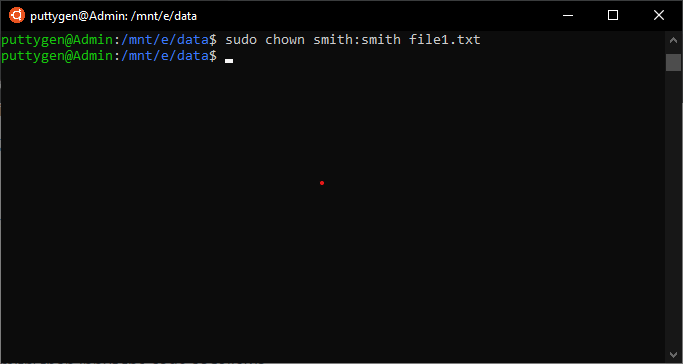
For example, we have to compare two files named - smith1.txt and smith2.txt.

The command will look like as follows:

diff -y -W 70 smith2.txt smith2.txt --suppress-common-lines

**8. chown Command**

The chown command is frequently used and is very simple to execute. It allows users to change the owner and the group owner of a particular file. Let’s first list the file using the ls -l that will describe the owner and group owner of the file.

~work$ ls -l

The code will list files with owners and group owners.

-rwxrw-w-1 smith smith 7655 Apr 25 17:26 trial.txt

In this example, ‘smith smith’ is the owner and group owner, respectively. All users have a default group, and it also indicates that the file is not shared with any other group. Using **chown**one can change the file owner and even the owner of a group. Coders have to use **sudo** to change or retain it back to the previous owner.

sudo chown smith:max trial.txt

Now to change even the owner and the group, input the following

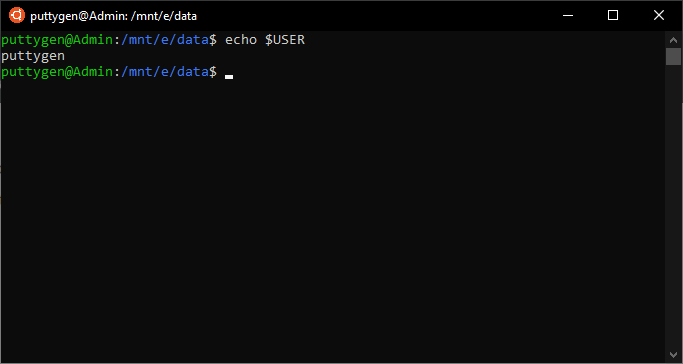
sudo chown max:max trial.txt

In case, user wants to revert to ‘smith’ than input the code as follows

sudo chown smith:smith trial.txt

**9. echo Command**

The echo command is leveraged to print a string of text passed as an argument to the terminal window. It comes as the built-in command and is frequently used get output status text in shell scripts. For example, a user wants to print ‘play with a string of text’ in the terminal, then it will look something like this:

echo play with a string of text

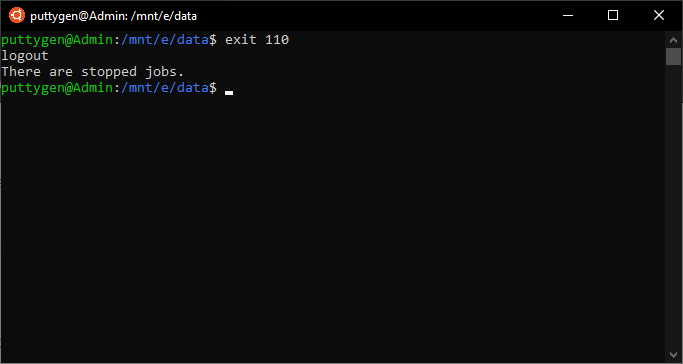
More so, one can even use echo to print the value of environment variables such as user and path of a file. For instance, if you want to know the user, home directory, or path, then enter the following command:

~$ echo $USER~$ echo $HOME~$ echo $PATH

The echo is also one of the most essential and crucial commands when it comes to shell scripts. Programmers can use the command with a script to print or display the progress or status of a script while it is running.

**10. exit Command**

The exit is the most basic command of all. All it does is exit the shell in which it is active, close the terminal and even logs out of an SSH remote session. Coders can use the exit command with or without parameters. So to close the terminal users has to enter the following command:

~$ exit

Now, let's look at an example with a parameter. For instance, users have to enter the following command to use exit with the help parameter:

~$ exit --help

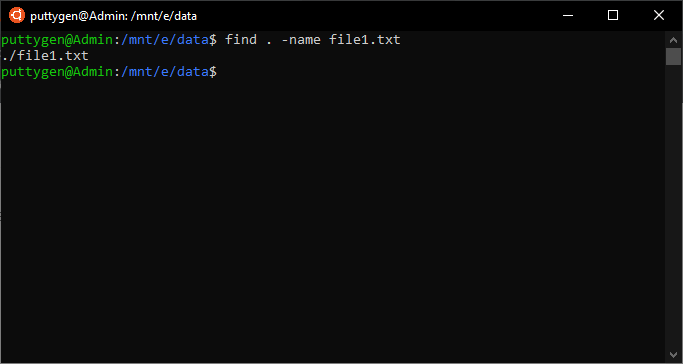
Coders should also use it with return status to ensure there are no errors at the time of closing the file. So the user can type the following command:

~$ exit 110

If the return value is ‘0’, then the script or program is executed without errors. Meanwhile, if the return value is ‘1’ then it the program consists of some mistakes.

**11. find Command**

The [find command](https://www.puttygen.com/find-command-in-linux) is one of the most useful commands on Linux. It is common not to know the location of files, and therefore, coders can run the find command to look for it. The only prerequisite from the user is to type in what to look for and where. More so, you can also use parameters to make the search more specific. Enter the following command to find a file:

~/Documents$ find . -name \*smith\*

The above command tells the terminal to find files that match the pattern. The \* looks correspond to the sequence of letters or characters. However, if a user wants to represent a single letter than they can use **? (question mark character)**instead of \*.

The above command will also include directories. In case you only want to look for only files, then programmers have to use the -type option. Let’s look at the command:

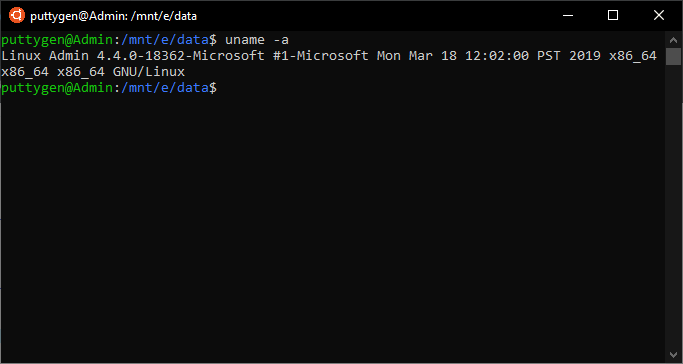
~/Documents$ find . -type f -name \*smith\*

The ‘f’ represents the files.

Users also have the option not to make the search case sensitive. Users have to enter the command as follows:

~/Documents$ find . -type f -iname \*smith\*

**12. uname Command**

The uname command provides users with the system information of the Linux system or computer they are using. It comes with several options that enable users to view all or specific information they want. Below is the option programmers can use with the uname command: 

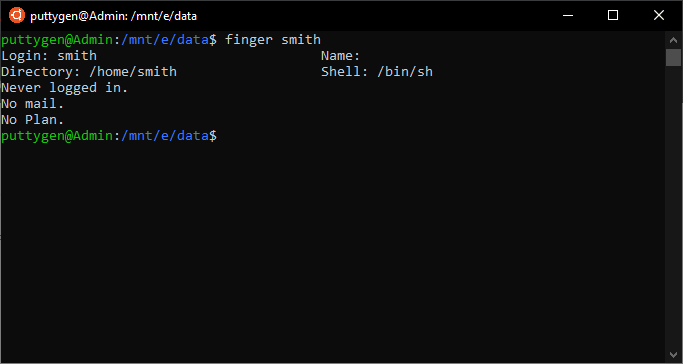
* -a = you can use this option to see everything
* -s = with this option users can view the kernel information
* -v = the option allows one to see the kernel version of the system
* -r = the parameter lets one view the kernel release

Therefore the command looks something as follows:

uname -auname -suname -v

**13. finger Command**

The finger command is leveraged b y programmers to look up and obtain brief information on users. It presents various data such as full name of the user account, real name, latest login information, idle time, write status, office location, and user’s home directory.



It also concatenates with multiple options such as **-s, -l, -p, and -m.** With the -s option, one can view all the information about a user mentioned above. However, using the -l, presents information about login shell, contents of file and mail status, in addition to all data provided with -s option.

The -p option is used to obtain information about the contents of the "**.plan," ".project," and ".pgpkey" files.**

Meanwhile, the -m option stops from matching usernames. Users must remember that the matching by finger command is case sensitive.

Below are basic examples using finger command:

~$ finger smith

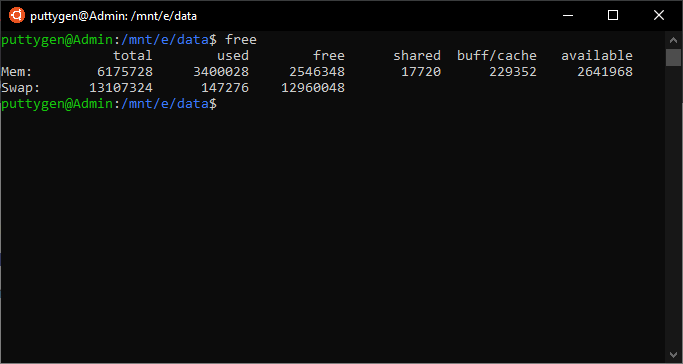
The command displays information about the user ‘smith.’

Now to view the contents of a file of the user, programmers can use the following command:

~$ finger -p smith

**14. free Command**

The free command-line utility provides users with a summary of the total available free space on the computer. More so, it also displays the total amount of used memory of the Random Access Memory (RAM) and the swap memory on the computer. Using the free command will also present the buffers leveraged by the kernel.

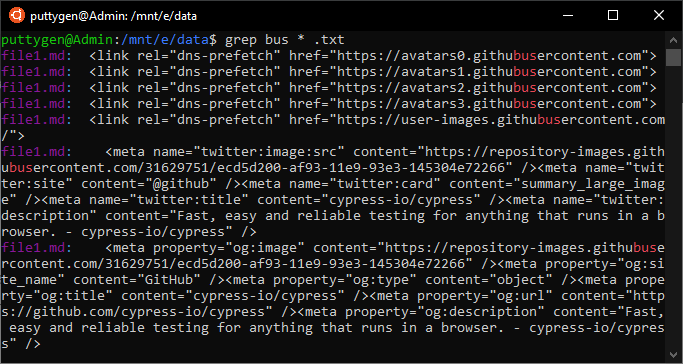


The output of the command displays information in bytes. However, coders can use the -h option to ensure the data is provided in human-friendly numbers. Values. Here is the example of the free command:

~/Documents$ free -h

**15. grep Command**

The grep command is one of the most useful commands on Linux. It is an acronym, and its full form is; global regular expression print.’ The sheer adequacy makes it a frequently used command on Linux. The grep command looks for lines and prints the ones that match a predefined search pattern.



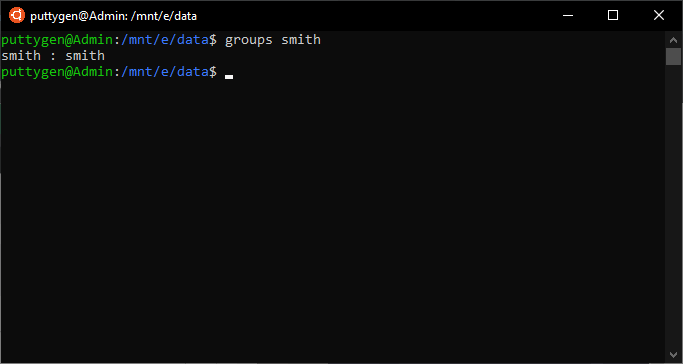
Commonly, most programmers use grep command to search for a particular string of words in a given file. More so, you can use the program to search for contents of the files. So for example, you have to enter the following command to look for the word ‘bus’ in all text files in the active directory:

~/Documents$ grep bus \*.txt

The result will show list not only the name of the file but also display the line it matches. The matching text is highlighted, making it easy for the user to look for the file on the computer.

**16. groups Command**

Linux has to mechanism two manage users - users and groups. In simple words, the groups are a collection of users. A group makes it easier to manage user with same permission privileges.



The groups' command is as simple as it sounds. It provides information about groups that a user is a member of. In simple words, when the command is executed, it will display the name of primary and supplementary groups for a username. The simple command looks like this:

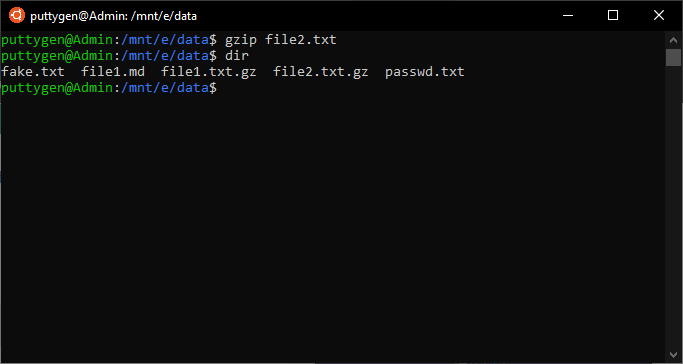
~/Documents$ groups smith

In case, the user does not specify a username; then it will print group membership for the active or current user. More so, for better understanding, you can also use the **--help command** with groups. So it will look something like this:

groups --help

**17. gzip Command**

The gzip is a useful command when it comes to file management. It is a file compressing tool with its primary goal is to reduce file size. The gzip command by default deletes the original file and saves only the new compressed file which is resulted from the command.



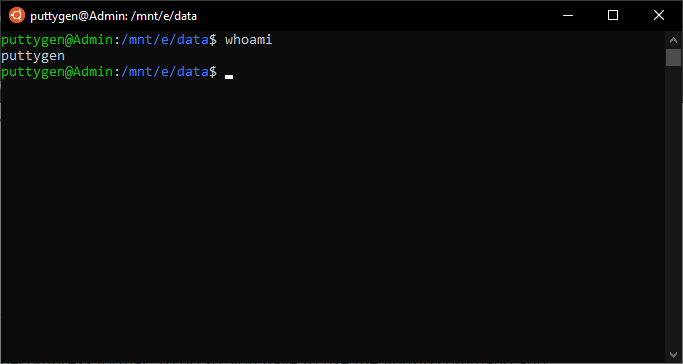
However, you can use the -k (keep) option to ensure that the original file is not deleted. The gzip command will look as follows:

~/Documents$ gzip -k example.c

Users can even revert the file to decompress format. To do so, you have to use the gunzip command and the original file will available at the location.

**18. whoami Command**

The whoami command is simple and mostly comes handy for amateur Linux users. It tells the user with a username they are logged in as. More so, coders can use the command to know if anyone has logged into an unnamed Linux terminal. It is straightforward to execute and is one of the few commands in Linux that perform a single task.

~/Documents$ whoami

The output will show the username of the one that is logged in as. Apart from Linux and other Unix-based operating systems, the command is also standard for the Windows operating system. Coders can use the command with two options help (--help) and version (--version). Similar commands include the**‘w’ and ‘who.’**

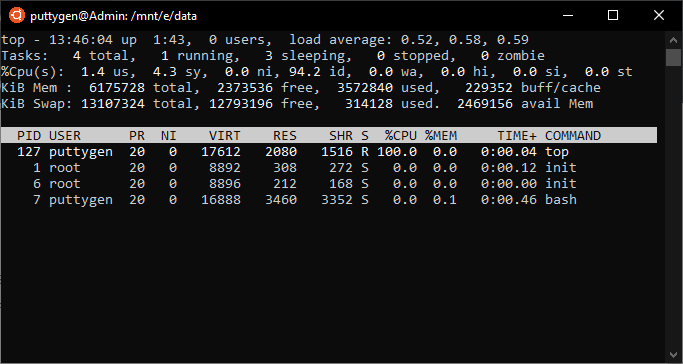
~$ w

Users can combine it with multiple options to filter users on the Linux system. For example, if you want information on a specified user, then programmers have to enter the following command

~$ w user

**19. top Command**

The top command is a task management program that presents a real-time display of CPU and memory usage of the Linux computer. It is a simple status summary. The command is suitable for those who like to monitor the CPU performance of the Linux machine. The user only has to type ***‘top’***in the terminal to run it.



It displays the number of users that are logged in, the runtime of the computer and the load average for various time intervals such as five and fifteen minutes. It also provides a summary of the task. The overview consists of the current status of the task, such as if they are running or sleeping or stopped. Below are letters that define the statuses of the process:

* R: Running
* D: Uninterruptible sleep
* S: Sleeping
* T: stopped (often known as Traced)
* Z: Zombie

The command will present other information in a column format. Here is what each column stands for:

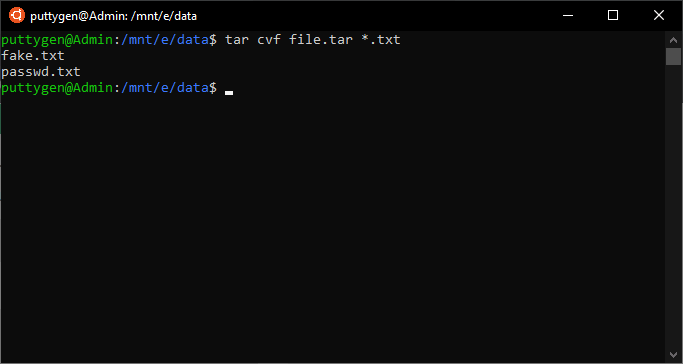
* PID: process ID
* USER: the Username of the owner of the process
* PR: priority of the process
* NI: the nice value of the process
* VIRT: this defines the amount of Virtual memory consumed by the process
* RES: resident memory ingested by that process process
* SHR: shared memory used
* S: it provides the status of the process
* %CPU: the amount of CPU time consumed by the process
* %MEM: the share of physical memory leveraged
* TIME+: composite CPU time used by the task

By default, the digits do not come in a human-readable format, but users can press ‘E’ to convert it to human-friendly values.

**Press ‘Q’**to exit from the tool after reviewing all information.

**20. tar Command**

The tar command is one of the most useful accounts when it comes to file management. Using tar, you can create archive files that can perform as storage for multiple files. The archive file is often known as the **tarball**.



Using tar makes file management convenience as one can share a collection of files quickly. You can even use the command to extract files from archives and also compress the archive. User must demand compress files otherwise the command will archive file in an uncompressed version.

Users have to define what files will the archive file have and the name that a user wants to give. For instance, let's say we want to create an archive for the Football directory, then enter the following command:

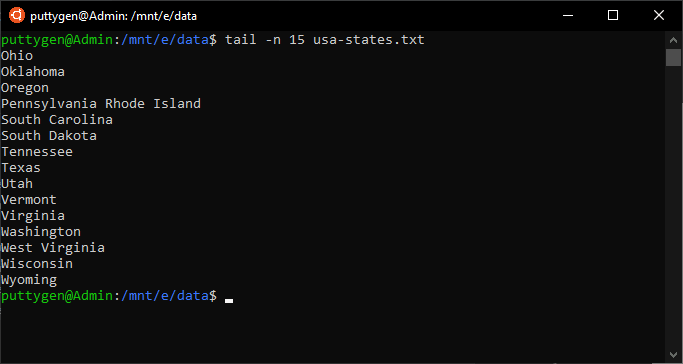
tar -cvf videos.tar Football/

In the example, the **-c** is the option to create, the **-v** is the verbose option that provides feedback while they are added to the archive.

Meanwhile, the -f is the filename that coders want to give to the archive. In this case, it is **videos.tar.**

**21. tail Command**

The tail command is a command-line utility that lists the last lines of a file that are given via standard input. The number of output lines is set to 10 by default. However, users can change it by combining the tail command with the -n (number) option. Programmers can use tail even to track files in real-time. It means that a user can view new lines as they are being written. Let’s look at the tail command as it is in default:

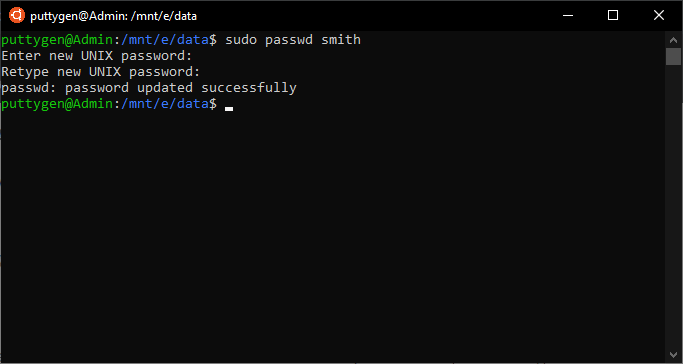
~/Documents$ tail example.txt

Now in case a programmer only wants to view the latest five lines, then one has to enter the following command:

~/Documents$ tail -n 5 example.txt

**22. sudo Command**

The Sudo command is one of the most advanced Linux commands. While the syntax is basic, but dealing with it is critical as it requires root access. The most common use of the sudo command is to change the password for other users. For example, if Smith wants to change the password for Max, then the command will be as follows

~/Documents$ sudo passwd max

The terminal will prompt Smith to input his password and then allow the change. The reason one must be cautious is that it will enable you to operate programs using the security privileges of another user. Coders can use the -l (list) option to know which commands is permitted and not permitted on the current host.

~$ sudo -l

**23. SSH Command**

The SSH command allows users to connect to remote Linux machines. You can even log into your account on the remote computer using the command. User has to provide a username, domain name, [IP address](https://en.wikipedia.org/wiki/IP_address) of the remote machine to make the connection. Let’s look at the example below:

~$ ssh Max@192.168.3.82

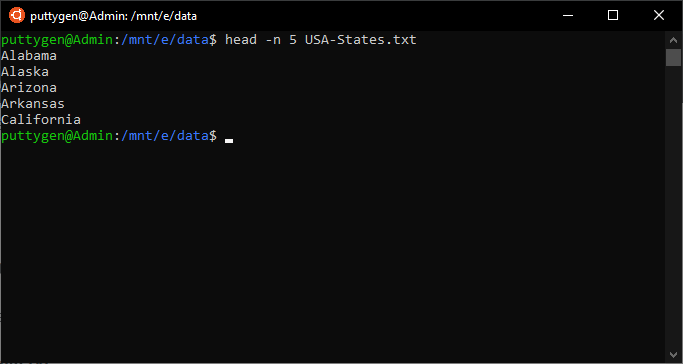
Using the command, Max is connecting to a remote machine whose IP is 19.168.3.82. It will demand to enter the password, and once verified and logged in; then the prompt will change to a system that the programmer is using. If Max writes the ‘w’ command, then it will list all the users of the system. As Max is connected via a remote machine, the connection type will state as **‘pts/1**’. The **pts/1** is a pseudo-terminal slave that displays that it is not directly connected to the actual computer.

To logout from the remote system, you have to type the exit command.

~$ exit

**24. head Command**

The head command provides users with the first lines of the file. By default, the output is set to the first 10 lines. However, users can alter it by using the -n (number) option. The head is the opposite of the command, which provides the last 10 lines of a file. In the first example, we will type the default command.

~/Documents$ head -example.txt

Now, let's see the command if we want to limit the output to only the first 4 lines.

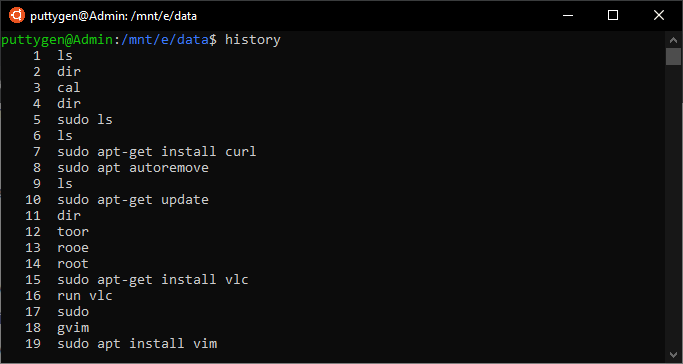
~/Documents$ head -n 4 example.txt

The command is standard among many of the Unix-line operating systems. The command also displays piped data. You can also use the head command to print the first 10 lines of multiple files.

head -n 5 example.txt example2.txt

**25. history Command**

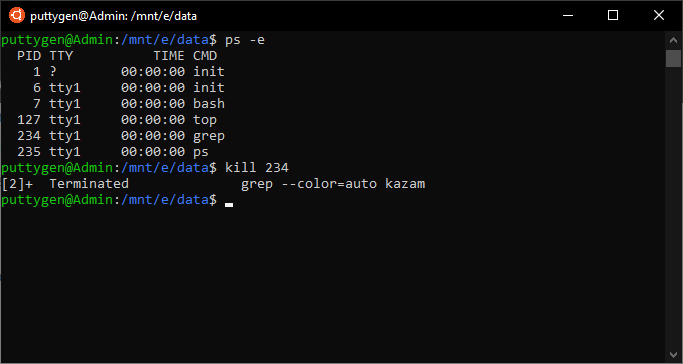
The history command is a way to view the commands that a user has inputted previously on the command line. The default limit is set to display only the latest five hundred commands. It is an easy way to repeat the commands just by inputting **! (exclamation)** the point with the number of the command that the user wants to repeat from the list.

~/Documents$ !180

The above will repeat the 180th command from the list. It is very useful because a user can avoid typing the command again. In case you want to repeat the last inputted command than one has to just input **‘!!’ points**. Coders can also look at the command history from the .bash\_history in your folder.

**26. kill Command**

The kill command offers the liberty to end the process from the command line. It is useful for those monitoring CPU processes as it makes it easy to terminate processes without working on it manually.



The user is only required to enter the Process ID(PID). Users must be cautious not to kill process randomly or haphazardly. Only use the kill command if the process or task must terminate or has locked up. To obtain the process ID, users have to use the ps and grep command.

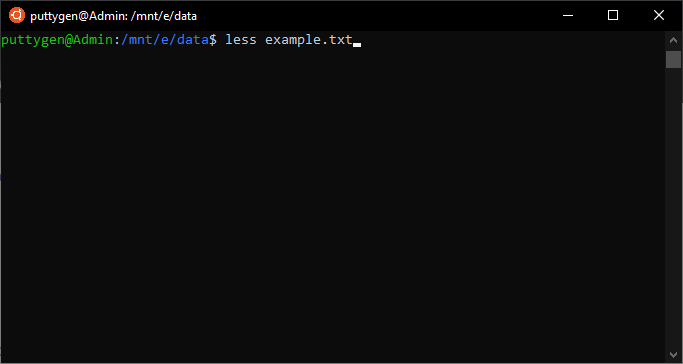
ps -e | grep Kazam

The command will display the Process ID; in this case, it is 1702. Now that we have the Process ID, let's kill the Kazam screenshot program.

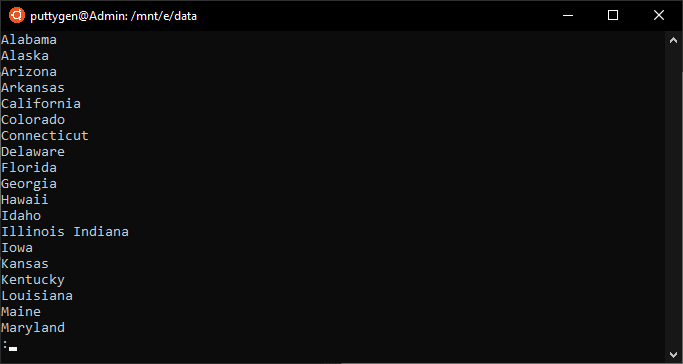
~$ kill 1702

**27. less Command**

With the less command, one does not have to use an editor to view files. It allows users to view files without fearing it being modified. The process is faster and best for those who believe their files might mistakenly be edited.



Output of above command



Once the file opens, users can view the entire document using the Up and Down key on the keyboard. Programmers can also make use of the PgUp and PgDn keys. Once viewed, users can quit less by pressing the**‘Q.’**Below is an example of the less command:

less example.txt

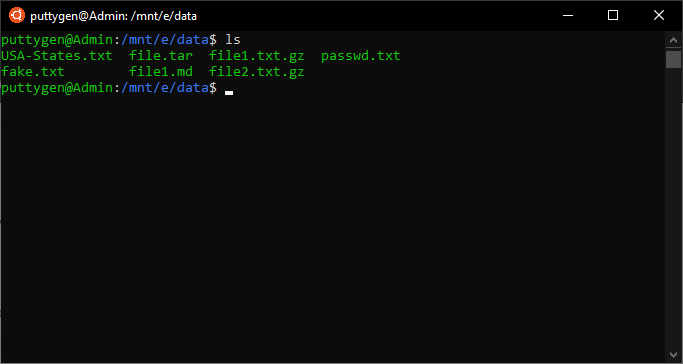
Coders can also use the less command from a list of files on the entire hard drive.

ls -R / | less

To navigate forward press ‘/’ key and to move backward press ‘?’.

**28. ls Command**

The ‘ls’ is the most basic and probably one of the first commands anyone learns. It is a simple command to list the directory of files and directories. The ls command will list files in the active directory if the user does not specify other. It is one of the most frequently used commands as users can concatenate its various options.

~/Documents$ ls

The above command will list all files in the current directory. For a comprehensive list, coders have to use the **-l(long)** option and use the **-h(human)** option to ensure the file sizes are mentioned in a human-readable format.

~/Documents$ ls -lh

In case, a user also wants to view hidden files, then enter -a (all files) parameter:

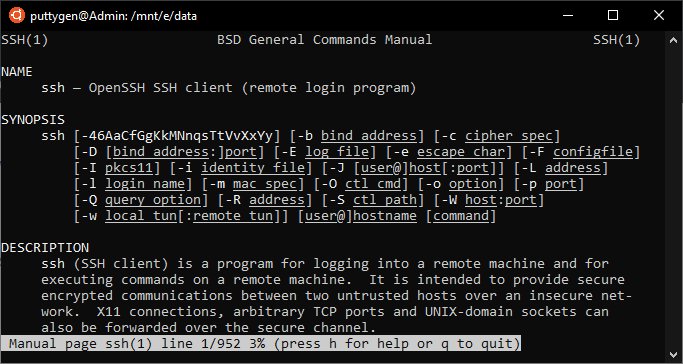
~/Documents$ ls -lha

To modify the list in order of latest updates, enter the following command:

~/Documents$ ls -lt

**29. man Command**

The man command displays the ‘man pages’ or user manual a command. It is advantageous as the programmer who is new to commands will have a better understanding before executing any Linux commands.



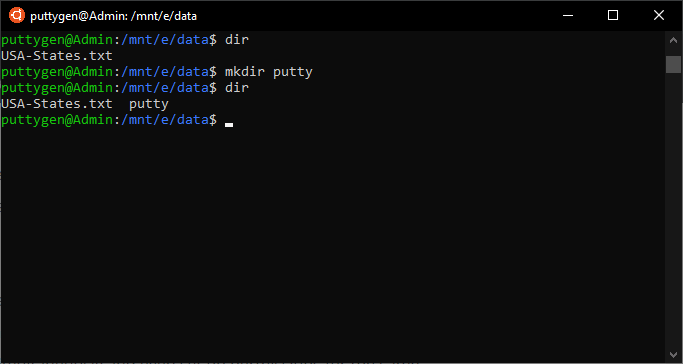
The man command provides a complete detail such as Name, return values files, examples, authors, and more. Users can leverage the capabilities of the less command as the ‘man’ command uses it to present the manual. For instance, to view the manual of SSH command type the following:

man SSH

Navigate the manual using the PgUp and PgDn keys. In case a programmer can’t understand the document than they have to press ‘h’ for help. Press ‘q’ to quit from the manuals.

**30. mkdir Command**

Using the mkdir command, coders are able to create new directories. Many operating systems refer to directories as folders. A coder only has to provide the name of the directory, and mkdir will create it. It can also create multiple directories simultaneously and even set up permissions for the same.



Programmers must also know that they have to provide a path for the new directory in case it is going to be placed in another directory than the current one. Let’s create a new file named **‘examples.’**

~/Documents$ mkdir examples

Now if the user wants to create a new directory dubbed ‘2020’ within the ‘examples,’ then enter the following command:

~/Documents$ mkdir examples/2020

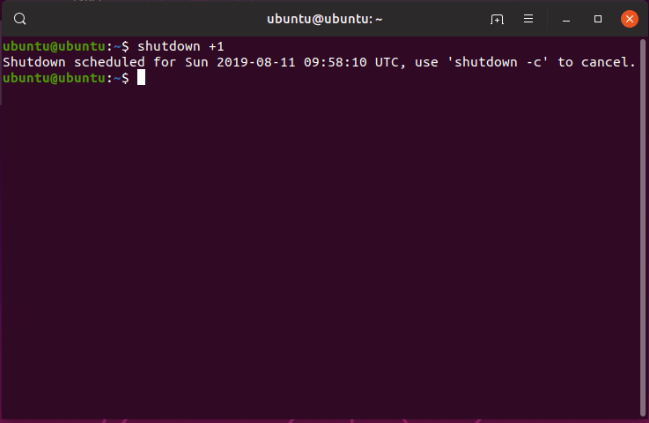
With mkdir, users can create a directory without the existence of its parent directory. To do so, you have to use **-p(parents) option**. For example, you want football as the parent directory of **‘2020,’** but it does not exist, then one can execute the command as follows:

~/Documents$ mkdir -p examples/football/2020

Even though ‘football’ does not exist, the mkdir command will create all of it.

**31. shutdown Command**

The shutdown command is simple as it sounds and most users will come across as soon as they learn about Linux commands. As the name sounds, the shutdown command will switch off or shut down your Linux system.



There are various parameters you can use with the shutdown command, but if no parameter is specified, then the computer will shut down in one minute.

~/Documents$ shutdown

In case you are in a rush and wants to shut down the computer immediately, then use the**‘now parameter.’**

~/Documents$ shutdown now

Users can even schedule shutdown and also notify all users for the same with a single command. The only thing users are to provide is the time and the pc will shutdown as per scheduled. You can also schedule in minutes, such as 10 minutes from now. Below are both examples:

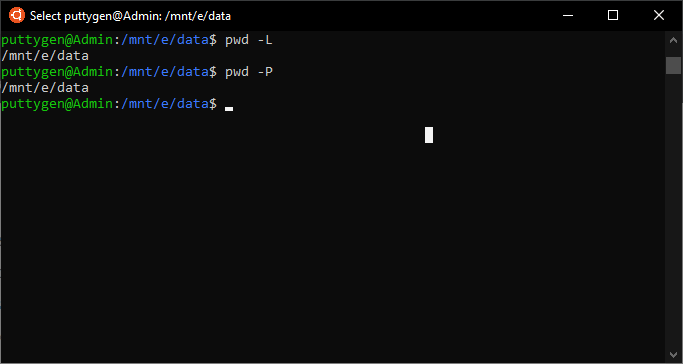
~/Documents$ shutdown tonight at 22:00, don’t forget to save your work and enjoy the weekend!~/Documents$ shutdown +10

Use the -**c(cancel) option** to cancel the scheduled shutdown.

~/Documents$ shutdown -c

**32. pwd Command**

The pwd command is simple as it can get. Its primary output is to print the active (in-use) directory commencing from the root/directory. It is an acronym and stands for Print Working Directory. The pwd command will print the full system path of the current directory. More so, $pwd is an environment variable, which means it will store the path of the active directory. It comes with two options, which are **-L and -P**.



* -L = it will print a symbolic path
* -P = it will print the actual full path

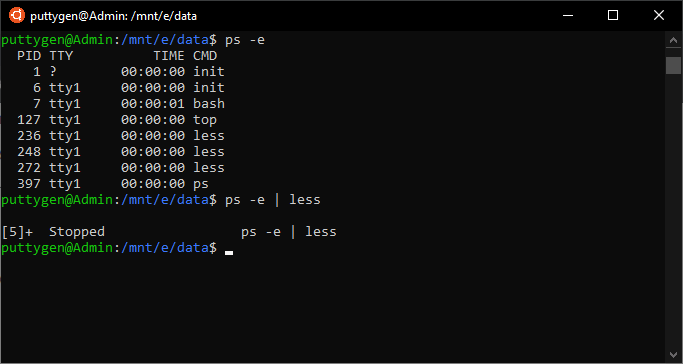
~/Documents$ pwd

To view the actual path, input the following command

~/Documents$ pwd -P

**33. ps Command**

The ps command displays a process that is currently running on the Linux machine. It provides a real-time display of all active processes. The ps command provides a basic summary of the process that includes its Process ID. In many operating systems, the process is referred to as tasks. By default, the ps command will list active operations in the current shell. However, you can use options to change to different shells.

~/Documents$ ps

In case, a user wants to view the process of a specific user then they can add -u(user) option to the ps command.

~/Documents$ ps -u smith

The list would be very long, and hence you can cut it short by adding the less command.

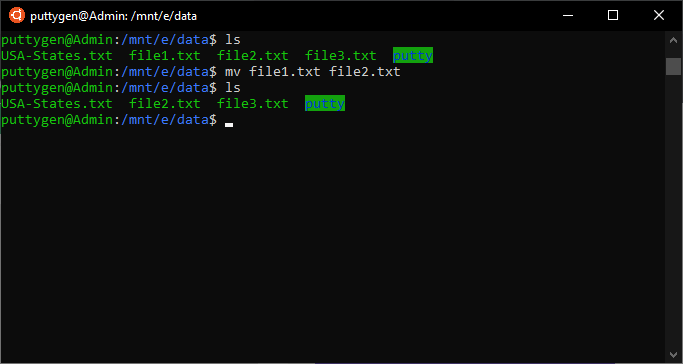
~/Documents$ ps -u smith | less

Use the -e(every process) to view every running process.

~/Documents$ ps -e | less

**34. mv Command**

The mv command enables programmers to shift files and directories between other directories. It is yet another file management command that makes organizing data easy and straightforward. Users can also use the command to rename files.



The only thing that the user has to define is the name of the file and where it needs to be moved. For example, let’s move a file dubbed ‘examples.pdf’ from ‘~/Document/Football’ directory to the current directory. The current directory is delineated using . Character.

~/work$ mv ~/Documents/Football/examples.pdf .

You can even rename file using the mv command.

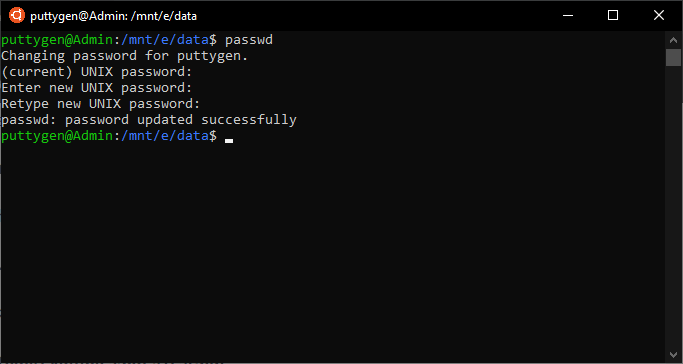
~/work$ mv examples.pdf test.pdf

Users can even achieve both tasks, i.e. moving the file and renaming it in a single command.

~/work$ mv ~/Documents/Football/examples.pdf ./test.pdf

**35. passwd Command**

With the passwd command, users can change the password for a user. It not only lets you change your password but also of other users. The passwd command is also one of the most frequently used Linux commands. The command comes in very handy as it makes it very easy to change the password for anyone in the team. To change others password, users have to use sudo command as it requires permission privileges. Type **paaswd** to change your own password

~/work$ passwd

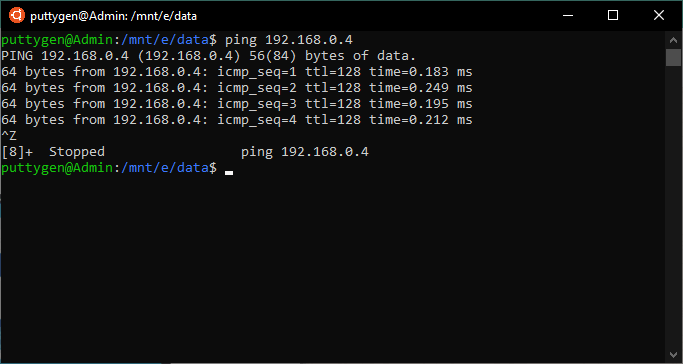
To change others password, type the following:

~/work$ sudo passwd max

It will demand the user to enter their own password. Once verified, the new passwords for other users.

**36. ping Command**

With ping command, users can verify if the network connectivity status between computer and source over an IP network. More so, you can view the response time from a network.



The response time is the time it takes to send and receive a response from the network. One of the most common uses of the ping command is to troubleshoot networking issues.

Coders have to provide an IP address to run the command. Below is how you can execute the ping command:

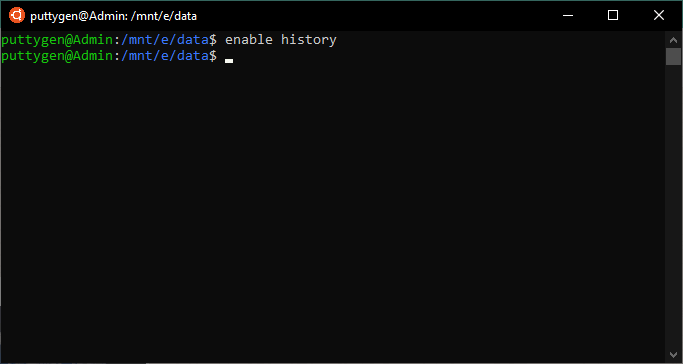
~/work$ ping 192.168.3.18

The ping runs unceasingly, and hence a user has to press Ctrl+C to stop it. However, you can use the -c(count) option to ping it for a specific number of attempts.

~/work$ ping -c 5 192.168.3.18

**37. Enable Command**

The enable command is a built-in shell command akin to the disable command. Coders use the enable to initiate the printers or classes, and similarly utilizes the disable option to cease the printers. Programmers can use the command with various options as listed below:



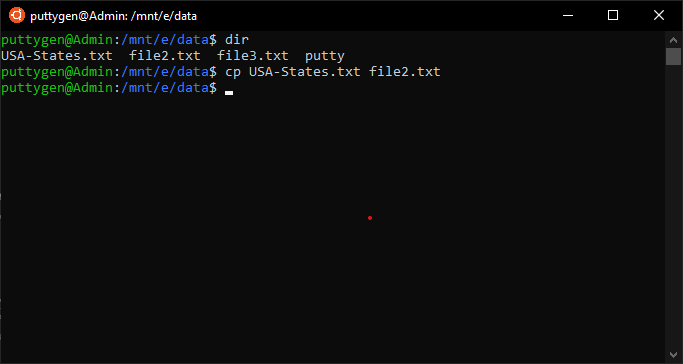
* **printer:** it refers to the printer that has to be enabled
* **-a:** it will list and display if built-ins are enabled or disabled.
* **-c:** the option terminates any active print requests
* **-d:** it will delete a builtin that has been loaded with
* **-W:** it will not disable printer and wait till the request is not printed. Programmers must note that they can’t use this and -c option together. The -W option will not work for remote printers.
* **-r reason:** the options will allocate a specific purpose to disable the printer.
* **-n:** by default names are enabled so that this option will disable it.

An example of the enable command will look as follows:

~$ Enable printit

**38. Cp Command**

The cp command is standard across many of the Unix-like operating systems. It stands for copy, and its primary function is to copy files and directories. Coders can even copy multiple files or directories using **cp.**



Each cp argument must have two filenames as it creates a mirror image of the file on a disk with a different name. The command-line utility can even move files or folders. Programmers can also choose to backups of files.

Look at the below code to copy a file using cp command:

~$ cp me.txt you.txt

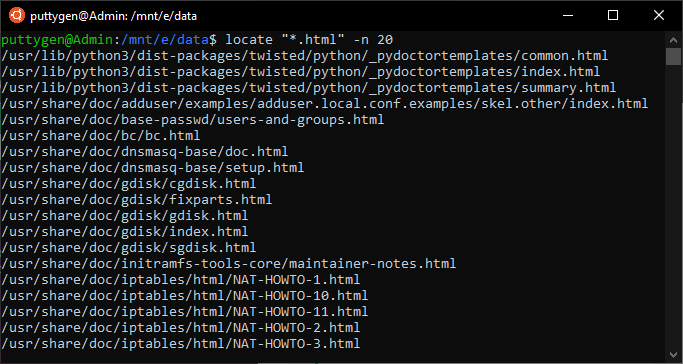
In the above example, the file me.txt will is copied to another file named you.txt. As mentioned above, the cp file will create a mirror file of the original file as part of the process.

Programmers can’t copy directories directly as it will end up in error. Therefore, they are required to pass the **-R flag**. The -R flag will, therefore, copy a folder and then create a copy. Let’s take as an example that football is the directory and one has to copy it. Input the following argument to do so:

~$ cp -R directory/ football

**39. Locate Command**

In Linux, the two most potent and widely used commands to search for files are ‘find and locate.’ While find is good without a doubt, the locate command is more powerful and presents results faster than the find command. Therefore, for instantaneous results, there is no better command-line utility than ‘locate.’



The reason locate command is faster is because it searches file through the database opposing to file command which initiates a search through the file system. The database has bits of files that result in an almost instantaneous result.

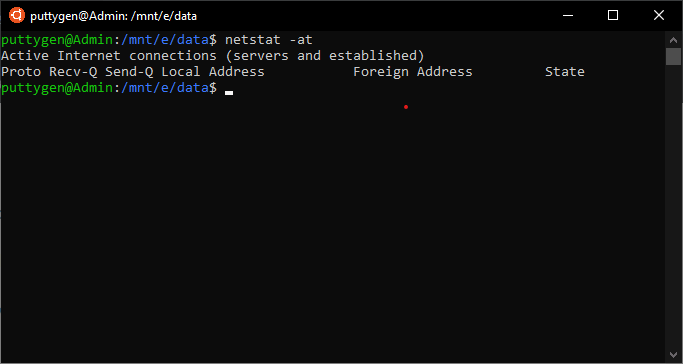
Executing the locate command is straightforward to use and one has to do is pass the name of the file. Let’s see it in action in the below example.

~$ locate me.txt

There are various options within the command, such as searching for matching entries.

**40. Netstat Command**

The netstat is a command-line tool that presents an overview of the network connections. Beyond that, the command also provides several network interface statistics, multicast memberships, routing tables, and masquerade connections.



Although it is standard on UNIX-like systems, many other operating systems also have netstat command.

Below are a few examples using the netstat command:

~$ netstat -a | more

The command will display listening and non-listening sockets. Users can always use the -interface option to view only the interface that is down.

Programmers can use netstat to list specific ports such as only list tcp or UDP ports.

Enter the following to view **TCP ports:**

~$ netstat -at

To only view **UDP ports**, use the below argument

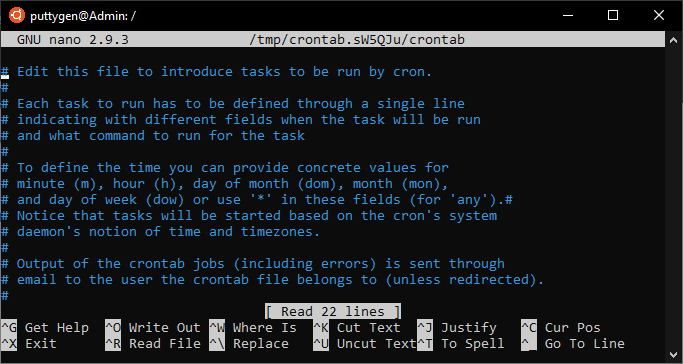
~$ netstat -au

You can even choose to view the statics of all ports. Input the following command display overview of all ports:

~$ netstat -s

**41. Cron Command**

The cron command is a software utility that is provided by Unix-like operating systems. The primary job of the utility is to schedule the task at a predefined time. Cron is a compelling and popular command that many uses to schedule task periodically at predetermined time or dates.



The time-based job scheduler automates scheduling making it very convenient. The cron runs in the background and is often referred to as a daemon process. Daemon process is a background process that runs for a long time and answers requests for services. The best feature of cron is that it performs its task without the intervention of a user.

Commonly most programmers use cron to accomplish repetitive tasks automatically. To complete this repetitive task is daunting, and hence, coders can schedule processes to run automatically. Users can even schedule processes such as organizing folders at given time intervals, taking backups at regular intervals, and also checking for updates.

Below are few basic cron commands:

**crontab -e** **=** the command runs crontab

**crontab -l =**you can edit crontab with this argument

**crontab -r =** it displays contents of the crontab

**sudo crontab -u -e =** using this command it will un-schedule all crontab tasks

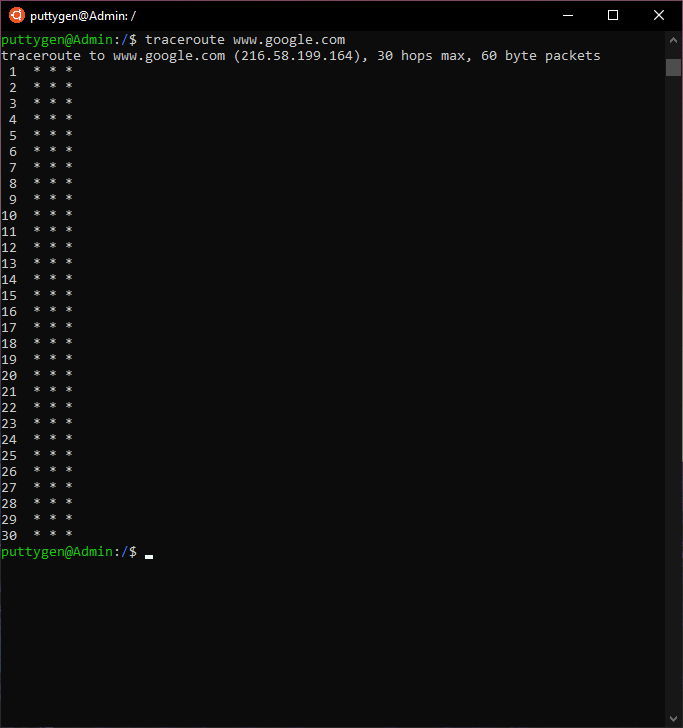
Let’s view an example whereby crontab takes a backup at a given time interval.

00 02 \* March Tuesday/home/max/backup.sh

The above command takes a backup of the folder every Tuesday in March at 2 AM.

**42. Traceroute Command**

The traceroute command tracks the route that a particular packet of information takes to reach to the host. In simple words, the command prints the journey that a packet takes from the source to destination.



One of the most common purposes to use traceroute is to track where data is lost in the network. As such, that coders can identify the particular node that is down.

To better understand, let’s track the route that a packet takes to reach Google from your computer. More so, the traceroute command will also display all jumps it makes during the journey.

Coders also have various options that they can use with the traceroute command.

~$ traceroute -4 10 google.com

The above command uses the IPv4. Meanwhile, the below command uses IPv6.

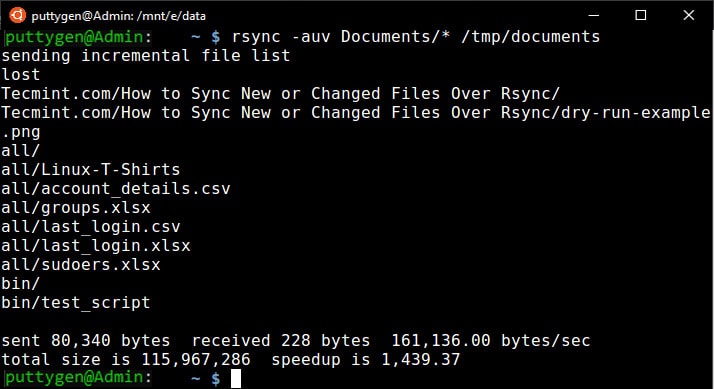
~$ traceroute -6 10 google.com

Users can also use the **-g(gate) option** to route the packet through the gateway.

~$ traceroute -g 192.168.43.45 google.com

**43. rsync Command**

Rsync stands for Remote Sync is a very standard command leveraged to synchronize files and directories. The software utility is usual among many Unix-like operating systems. The primary purpose of the utility is to sync files between hosts or machines, both locally and remotely. Beyond that, Rsync is also used to copy files.



The rsync utility can work in two ways sync data, which are as follows:

* Sync to/from another hose using remote shells such as SSH.
* Sync via Rsync daemon leveraging TCP.

Programmers can even use various options with the Rsync command

* **v:**verbose
* **-z:** it compresses file data
* **-r:** the option allows users to copy data recursively, but it does not save permissions.
* **-a:** with this option, users can copy files in the archive mode and even save permissions and timestamps.
* **-h:**the option offer data in a human-readable format

Below is a basic example of transferring files between two local machines. In the example, we copy a file dubbed examples.tar to /tmp/football/ folder.

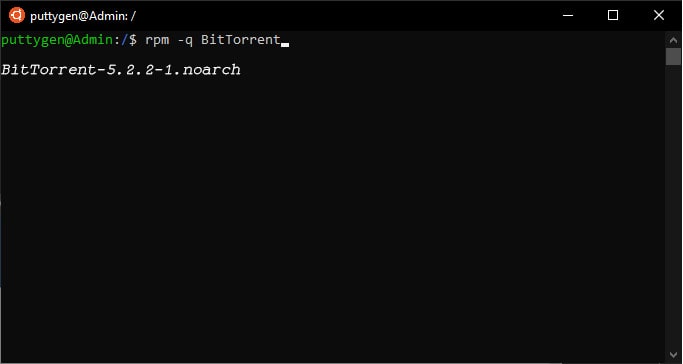
~$ rsync -zvh examples.tar /tmp/football/

Below is the argument to copy/sync a folder - test, to or from a server:

~$ rsync -avz test/ root@192.168.1.105:/home/

**44. RPM Command**

RPM stands for Red Hat Package Manager, which is a tool that permits users to install, verify, manage, and uninstall software packages in Linux and Unix-like operating systems. However, programmers must remember that the utility only works if the software is developed using the **.rpm format.**



The tool has five basic modes, which are as follows:

* **Install:** it install RPM packages
* **Remove:** coders can use this mode to uninstall or remove RPM packages
* **Upgrade:** with this mode, one can update RPM packages available on the system
* **Verify:** coders can use to verify any RPM package
* **Query:** it is commonly used to query any RPM package

The command below will list all RPM packages on your Linux machine:

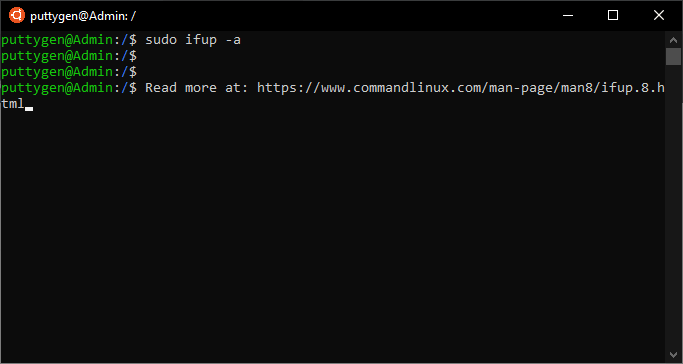
~$ rpm -qa

Users can enter the following argument to see if RPM is installed or not on a package. Coders have to use the **-q option.**

~$ rpm -q BitTorrent

**45. Ifup Command**

The ifup command is an essential and straightforward way to bring the network interface up and enable it to transfer and receive data. In other words, coders use the ifup command to configure network interfaces. Programmers must take note that the command is used with Sudo as it deals with permission privileges.



For example, to get the network interface up to use this command.

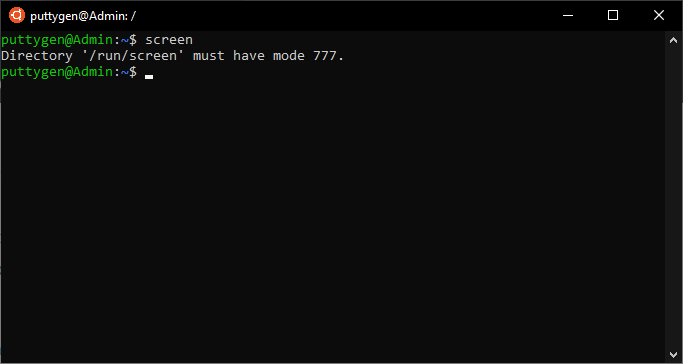
~$ sudo ifup -av

Users can use the below options with the command:

* **-a(–all):** if coders use this option then they must note that it will apply to all interfaces labeled as auto.
* **–force:** the option permits users to force configuration of the interface.
* **-V:** the option prints the information of the version in use
* **-v:** verbose the result as they are performed

**46. Screen Command**

The screen command permits users to initiate multiple shell sessions via a single ssh session. The best feature of the screen command is that the process can be detached from a session and then join the session again at a later period. It is possible as the process because the screen runs the process that was started even after being detached.



Therefore, making it possible to reattach the session at a later time. So one can begin the same place where they left off in the terminal.

The below command will install ‘screen’:

~$ sudo apt install screen

Typing ‘screen’ will launch a window within the screen.

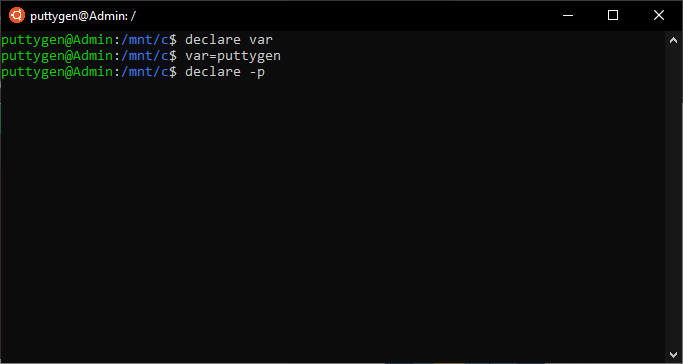
~$ screen

The command will initiate a new window within the screen and also give the windows it a name.

~$ screen -s test

**47. Declare Command**

The declare is a bash shell builtin command which means that it is a part of the shell. It has multiple purposes, such as declare shell functions, display variables, and more.



Coders can use the command with various options listed below:

* **-p:** the option shoes options and attributes of every variable name that used name arguments.
* **-f:** it is considered as the name of the function and not as a variable.
* **-F:** using this option, coders can choose not to view the contents of the function but rather only the attributes and function's name.

The below example shows how to declare a variable:

~$ declare var

Coders can use the command to view the variable name of each attribute:

~$ declare -p

**48. Cpio Command**

The cpio command is a standard among many Unix-like operating systems with the primary purpose being processing archive files. In simple words, programmers can use the cpio command to copy files from archives such as **.cpio or .tar.**



Let’s view an example to extract a .cpio file. The command will extract files and directory with a single argument.

~$ cpio -iv < /document/linux/example.cpio

Programmers can even create .tar files using the cpio command as below:

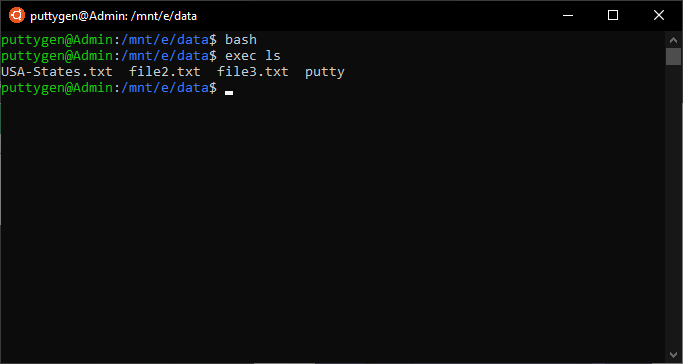
~$ cpio -ov -H tar > /document/linux/examplefile.rar

Similarly, coders can even use cpio to extract .tar files. Below is an example of how to extract the ‘examplefile.rar.’

~$ cpio -iv -F /document/linux/examplefile.rar

**49. Exec Command**

The exec command is leveraged to run a command directly from the bash. Therefore it does create a new process but instead substitutes it with the command that has to be executed. So, if the argument is successful, then it won't come back to the calling process.



Programmers can use the exec with various options listed below:

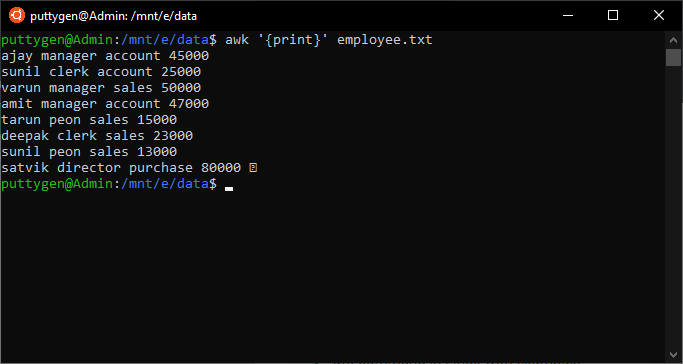
* **c:** users can execute a command with an empty environment
* **a name:** programmers can use the option to pass a name
* **l:** it passes dash as the zeroth argument of the command

The exec can be used as an argument or without a command. Below is an example of the exec:

~$ bash~$ exec ls

**50. Awk Command**

Awk is a software utility that one can leverage to write small programs in the way of statements. Users can utilize these statements to define text patterns which can be searched in a document. One of the primary purposes of Awk is to scan patterns and processes.



Awk is actually a scripting language that is mostly leveraged by programmers to manipulate data. With Awk, coders do not have to use compiles but can run scripts with logical operators, variables, and string functions.

Below is an example of using awk. First, we use it to print all lines in a file.

~$ awk '{print}' team.txt

Now let’s try to give the expression pattern.

~$ awk '/strikers/ {print}' team.txt

Download Linux Commands PDF

We will shorty upload a linux commands pdf file to download for free. Meantime, you can bookmark this page for all linux commands.

The article covers most of the **basic Linux commands** that one will come across. There are various other commands, but commonly the above listed 50 Linux commands will do the job. More so, running commands on Linux is all about combining it with the right option. There are multiple ways to get the job done, and one must use logic to choose the best possible option.

[[](https://www.puttygen.com/find-command-in-linux)](https://www.puttygen.com/find-command-in-linux)

[Find Command in Linux With Examples](https://www.puttygen.com/find-command-in-linux)

[[](https://www.puttygen.com/putty-commands)](https://www.puttygen.com/putty-commands)

[Basic SSH (Putty) Commands](https://www.puttygen.com/putty-commands)

[[](https://www.puttygen.com/make-a-minecraft-server-using-ssh)](https://www.puttygen.com/make-a-minecraft-server-using-ssh)

[How to Make a Minecraft server using SSH](https://www.puttygen.com/make-a-minecraft-server-using-ssh)

© 2020 Puttygen.com

[Contact](https://www.puttygen.com/contact-us) [Privacy Policy](https://www.puttygen.com/privacy-policy) [Disclaimer](https://www.puttygen.com/disclaimer)